Heuristic Motion Planning with Movable Obstacles

Thomas Chadzelek Jens Eckstein Elmar Schömer *

22nd April 1996

Abstract

We present a heuristic approach to geometric path planning with movable obstacles. Treating movable obstacles as mobile robots leads to path planing problems with many degrees of freedom which are intractable. Our strategy avoids this computational complexity by decoupling the whole motion planning problem into a series of tractable problems, which are solved using known path planning algorithms. The individually computed solutions are then coordinated to a path plan. This method results in a powerful and practicable strategy for path planning with movable obstacles, which can be applied using a wide variety of known motion planning algorithms.

1 Introduction

Objects in geometric path planning problems are usually divided into *moving objects* and *fixed* ones called *obstacles*. In a problem description the moving objects and the obstacles are given - usually as polygons or polyhedra - together with their positions and orientations. For the moving objects goal configurations are specified additionally to complete the problem description.

Much work has been done to solve this kind of problems ($\longrightarrow$ classical path planning algorithm CPPA). In early papers it has been shown that the problem is decidable for an arbitrary number of moving objects [10]. In subsequent work the focus of interest has been on two classes of algorithms. Firstly *complete* algorithms have been developed which all convert the path planning problem for objects to a path planning problem for points in configuration space. They vary widely in the ability to handle different degrees of freedom, the representation of configuration space and the efficiency to compute it. Their common point is the ability to decide whether the given path planning problem is solvable with the feasible degrees of freedom. Examples for those algorithms are in [4, 5, 7]. They suffer from the major drawback that only small problems can be solved this way. The second class are *heuristic* algorithms [1] which compute the configuration space only partially and are able to handle more degrees of freedom or more complex scenes. But these algorithms cannot decide whether there is a solution to the given problem.

Both classes of algorithms do not take into account that usually the objects in a scene cannot be strictly divided into moving objects and fixed objects. In real world most obstacles are *not fixed* but *movable*. A human planner would take into account that he can move those obstacles in order to solve his problem. This third class of objects which do not have to but can be moved might theoretically be treated as moving objects. Typically there are many movable obstacles but only very few of them have to be moved to solve the path planning problem. Treating them as moving objects makes those problems intractable.

In spite of the observation of movable objects in reality only little research has been done on motion planning with movable obstacles. Wilfong presented an approach to this kind of problem in [12], but he considered only a 2-dimensional workspace und his focus was the grabbing and pushing of obstacles.

Our main interest is to detect, whether obstacles have to be moved in order to give way to the moving object and how these motions can be computed and coordinated to a path plan. For that we present a heuristic strategy which decomposes the complex problem into a series of tractable problems whose solutions are combined to an integrated solution of the whole problem. The strategy is applicable to 2- and 3-dimensional workspaces und uses four major steps:

1. Compute paths for the moving objects which are collision-free with regard to the fixed ob-

*Lehrstuhl Prof. G. Hotz, FB 14 Informatik, Universität des Saarlandes, Saarbrücken, Germany. Please contact our WWW-server `http://hamster.cs.uni-sb.de` for more information, including technical reports.

jects.

2. Determine the set of movable objects obstructing the previously computed paths.

3. Independently compute paths for the obstructing movable objects which clear the paths of the moving objects.

4. Coordinate the motions of the obstructing movable objects.

This decoupled planning strategy has firstly been developed in [2]. A similar idea for multiple moving objects has been used in [9].

2 Problem Definition and Results

Our problem can be specified as follows: Given a set $O^S = \{o_1^S, \ldots, o_n^S\}$ of static obstacles, a set $O^M = \{o_1^M, \ldots, o_m^M\}$ of movable obstacles and a moving object o^M with start configuration c_{start} and goal configuration c_{goal}, compute a collision-free motion p^M for o^M with respect to O^S and a motion plan MP, i.e. a sequence of coordinated collision-free (with respect to all objects in the scene) motions for objects in O^M, so that p^M becomes collision free considering $O^S \cup O^M$ after executing MP.

We present a general strategy which is applicable to a wide variety of CPPAs. It enlarges in a practical manner the abilities of the CPPA without using impracticable methods of path planning for multiple moving objects. The CPPA is only required to compute a path map without having a goal configuration. The path map is used to coordinate the motions of the different movable obstacles spatially.

The strategy can easily be adapted to path planning algorithms for

(a) dynamic environments and

(b) multiple moving objects

In the following we describe the four major steps of the strategy in more detail and illustrate them with examples.

3 The strategy

3.1 Path Planning for the Moving object

In a first step the movable obstacles are ignored and the classical path planning problem for the moving object is solved considering only the static obstacles.

For this problem any known CPPA can be used such as [4, 5, 7]. The only result relevant for future computation is the description of the path p^M for o^M.

3.2 Determining the Obstructing Movable Obstacles

With the path from step 3.1 we could now use standard collision detection techniques to determine the obstructing movable obstacles. The result would be a set $O_{col}^M \subseteq O^M$ of colliding movable obstacles, but we would not have any information on where to place the obstacles from O_{col}^M in order to clear the path for o^M.

To get this type of information we take a closer look at the result of step 3.1. The moving obstacle together with the motion form a 4-dimensional object-time-space. A collision-free path for o^M with respect to O^M is equivalent to the fact that no obstacle from O^M has a non-empty intersection with o^M at any time of the path. So we can use the 3-dimensional projection of object-time-space on the 3-dimensional object-space called *swept volume* to compute the obstructing movable obstacles. Furthermore we get a representation of the region in object-space which has to be cleared and this can be used to efficiently determine goal positions for the obstructing movable obstacles.

Computing the exact swept volume for all type of possible movements (pure translations, pure rotations, mixed translations/rotations) is difficult and not very helpful, because most of the intersection detection algorithms in computational geometry are based on a polyhedral boundary representation of the objects. The swept volume forms only in the case of a pure translation a polyhedral region, pure rotations create hyperbolic surfaces and mixed translations/rotations, free formed surfaces.

So we compute only for pure translations the exact swept volume and for rotations and mixed translations/rotations we approximate the exact swept volume by a polyhedron which safely includes the real swept volume. For details on the computation techniques refer to [2].

The polyhedral representation of the swept volume can now be used to determine the set $O_{col}^M \subseteq O^M$ with standard polyhedron intersection algorithms (see [11]).

The result of this step is the set $O_{col}^M \subseteq O^M$ and the polyhedral representation of the volume swept by o^M.

3.3 Computation and Coordination of Clearing Paths

The set O^M_{col} is the set of movable obstacles which definitely have to be moved to clear the precomputed path for o^M. Although O^M_{col} is usually only a small part of all movable obstacles and the computational complexity is substantially reduced, it is usually big enough to make a complete simultaneous computation and coordination of clearing paths for all obstacles in O^M_{col} intractable. That's why we also use a decoupled planning strategy for this problem in order to make the remaining computational complexity tractable. The result is a motion planning problem without goal configuration for the moving object. But we have a polyhedral region which has to be cleared by the moving obstacles, i.e. the complement of the swept volume determines the region where each moving obstacle has to be completely in at the end of its clearing path.

3.3.1 Determination of Goal Configurations

Most of the known motion planning algorithms compute either a direct representation of the configuration space (C-space) for the moving object as [7] or discretizations of C-space like visibility graphs, Voronoi-diagrams or probabilistic roadmaps.

In case of a direct representation of C-space the algorithms are able to compute the intersection of different regions so that the free configuration space (free-space) and the free configuration space concerning only the polyhedral approximation of the swept volume as obstacles (free-S-space) can be computed separately and the intersection of free-space and the free-S-space form the set of all *possible free goal configurations* of the obstructing movable obstacle. In this region either randomly chosen points or special sample points can be used as goal configurations for the classical motion planning algorithm. In case of a discretization of C-space standard polyhedron intersection algorithms can be used to classify points in C-space as *possible free goal configurations.*

The computation and coordination of clearing paths can be done in two structurally different ways:

(1) Heuristic computation and coordination in one single step

(2) Coordination after computation of independent clearing paths

The first is the faster but less powerful possibility to solve the path clearing problem, the second requires more computational effort but can solve more kinds of problems.

3.3.2 Heuristic Computation and Coordination in One Single Step

Besides complete simultaneous computation and coordination of clearing paths which is known to be intractable there are some substantially faster but less powerful strategies to solve the clearing path problem:

(1) *Known heuristic motion planning algorithms for multiple moving objects* as in [3] which can be modified concerning the goal configurations of the objects

(2) *A priori motion order*
This order can be intrinsic to the scene or chosen interactively or randomly. It creates a sequence of classical motion planning problems where all obstacles are static and in each stage the obstructing movable obstacles which have already moved are considered in their goal configuration whereas those which still have to be moved are considered in their original configuration.

The computed sequence of motions forms in canonical manner a path plan which clears the precomputed path for the moving object o^M.

This method is a simple straightforward method of motion coordination and succeeds in realistic scenes as shown in figure 1, but there are many situations where more sophisticated coordination schemes have to be used (see figures 2,3,4).

3.3.3 Coordination after Computation of Independent Clearing Paths

Known heuristic motion planning methods for multiple moving objects suffer from the major drawback that they are not able to revise computed paths efficiently. We overcome this problem by computing the paths independently and using the discretization of free-space (as visibility graph, voronoi diagram or probabilistic roadmap) to chronologically and spatially coordinate the different paths. This idea has also be suggested by Overmars (see [9]).

Computation of Clearing Paths

We do this by considering each obstructing movable obstacle by his own and compute an independent (of the rest of the obstructing movable obstacles) clearing path. Thereby we consider the set of static obstacles O^S, the non-obstructing movable obstacles $O^M \setminus O^M_{col}$ and the moving object o^M in start configuration as obstacles in the scene. This results in a classical motion planning problem. To coordinate the independently computed paths spatially the discretization of the free-space is also stored as a result besides the computed path.

Coordination of Clearing Paths

The paths are all independently computed, so that they have to be coordinated to build a collision-free path plan which clears the path of the moving object. The coordination can be realized as pure motion ordering, time coordination of paths or time and local coordination of paths. There are different ways to realize the coordination:

1. Motion ordering
2. Chronological coordination
3. Chronological and spatial coordination

Motion Ordering

To decide whether there is a motion order for the clearing paths we compute a directed graph called *ordering graph*.

Definition .1:
Given a set $S = \{s_1, \ldots, s_n\}$ of pairs (o_i^M, p_i) of moving objects o_i^M and clearing paths p_i, the ordering graph $OG = (V, E)$ is defined as follows:

$$\begin{aligned} V &= S \\ E &= \{(s_i, s_j) | o_i^M \textit{ has to be moved before } o_j^M\} \end{aligned}$$

Iff the ordering graph is acyclic, the motions of the o_i^M can be ordered collision-free by topologically sorting the graph. For each pair of objects (o_i^M, o_j^M) the edges of the ordering graph can be computed using standard collision detection techniques:
If o_j^M obstructs the path of o_i^M in its start configuration, then o_j^M has to be moved before o_i^M and hence an edge (o_j^M, o_i^M) is added to the graph. If o_j^M obstructs the path of o_i^M in the goal configuration, then o_i^M has to be moved before o_j^M and an edge (o_i^M, o_j^M) is added to the graph.
The topological ordering of the motions defines one possible clearing path plan.

Motion ordering is applicable to many realistic scenes as shown in figures 1 and 2, but there are situations where spatial coordination is required (see figure 4) or more sophisticated chronological coordination of paths is needed (see figure 3).

Chronological (and Spatial) Coordination

Pure motion ordering is only a straightforward method of motion coordination because the motion for each obstacle is performed at one time. This is sufficient for problems where the clearing paths do not interfere with each other very much. Generally a more sophisticated coordination strategy is required. We meet this demand by timely (and locally) coordinating the clearing paths. The discretization of free-space (if computed by the planning algorithm) is used to locally coordinate the clearing paths. The decomposition of the paths into path components is used to chronologically coordinate the paths. We introduce the datastructure of *coordination graph* which is a superset of Overmars' *super-graph* (see [9]):

Definition .2:
Given a set of simple roadmaps $R = \{(V_i, E_i) | i = 1, \ldots, n\}$, the corresponding coordination graph $CG = (V, E)$ is defined as follows:

$$\begin{aligned} V &= V_1 \times \ldots \times V_n \\ E &= \left\{ (v, w) \;\middle|\; \begin{array}{l} \forall i : (v_i, w_i) \in E_i \quad \wedge \\ \exists \textit{ coordinated motion for} \\ o_i^M \textit{ from } v_i \textit{ to } w_i \end{array} \right\} \end{aligned}$$

The definition is general concerning both the roadmap (simple path or complete discretization of free space) and the coordination scheme. Overmars suggested a coordination which allows only one object to move at one time. We present two more techniques here which serve the purpose of coordinating the clearing paths:

(1) *Multiple moving objects without chronological coordination*

A subset $S_M \subseteq S$ can be moved collision-free without velocity coordination. That means that the objects can be moved at one time or in any order. This can be computed using standard motion planning techniques (see [11]) or reusing the algorithm computing the swept volume of moving objects. A motion is possible if there is no intersection between any two swept volumes. (Note that the objects which do not move in one step have to be considered both using collision detection techniques as static objects and the swept volume technique with the object as swept volume.)

(2) *Multiple moving objects with chronological coordination*

Kant and Zucker presented in [6] a technique to heuristicly solve the motion planning problem with multiple moving objects. They divided the planning process in two major steps. In the first one they computed individually the paths for the moving objects and in the second they chronologically adapted the paths by computing a velocity profile for each object. The second step can here be used to compute a chronological coordination for the objects moving in this step.

To coordinate the moving objects it has firstly to be determined whether they can be moved individually without collision considering the objects which do not move in this step. After this the second step of Kant and Zucker's technique can be used to compute a velocity profile for each moving object to chronologically coordinate them.

Those techniques are able to solve more complicated coordination tasks, too (see figure 3 and 4).

Speeding up the Graph Search

With the technique that only one object can be moved at a time Overmars proved empirically that it is practicable to compute the whole coordination graph in advance. If the coordination schemes are more general, i.e. more than one object can be moved at a time with or without chronological or spatial coordination, the coordination graph can be very big and it may be intractable to compute the whole graph.

In this case the A^* algorithm can be used to scan the graph without computing it in advance (see [8]). As a criterion for the optimal solution the minimization of the sum of all path lengths is used. In order to find the optimal solution the A^* algorithm needs an approximation of the remaining path length which safely underestimates the real remaining path length. So we define a distance measure (e.g. euclidean distance) on the individual roadmaps of each moving object and we compute for each pair of nodes (v, w) the shortest distance $d(v, w)$ using e.g. the all-pair-shortest-path algorithm of Floyd and Warshall. Using this distance-matrix we can determine for each node v the minimal distance to a safe goal node $d_{min}(v) = min\{d(v, w) | w \in SGN(R)\}$, where $SGN(R)$ is the set of all safe goal nodes of a roadmap R. We combine these measures to an overall measure in the coordination graph using the sum of all individual measures. So the estimation of the remaining path-length for a node v of the coordination graph is

$$h(v) = \sum_{i=1}^{n} d_{min}(v_i)$$

This distance estimation takes into account that an object aims at varying goal positions when it is forced to avoid other objects during coordinated motion.

Coordination Path $\longrightarrow$ Path Plan

The computed coordination path now has to be interpreted as a path plan. Each edge of the coordination path stands for the movement of one or more object(s) along an edge of the individual roadmap which may be chronologically coordinated by velocity profiles. So the whole coordination path describes a collision-free path plan for the obstructing movable obstacles to clear the precomputed path for the moving object.

References

[1] T. Chadzelek *Heuristic Motion Planning with Many Degrees of Freedom* Technical Report A 08/95, University Saarland, 1995

[2] J. Eckstein *Heuristische Bewegungsplanungsstrategien im* $\mathbb{R}^3$ Master Thesis, University Saarland, 1994

[3] M. Erdmann, T. Lozano-Perez *On Multiple Moving Objects* Algorithmica, Vol. 2, pp. 477–521, 1987

[4] L.J. Guibas, M. Sharir, S. Sifrony *On the General Motion-Planning Problem with Two Degrees of Freedom* Discrete Computational Geometry, Vol. 4, pp. 491–521, 1989

[5] K. Kedem, M. Sharir *An Automatic Motion Planning System for a Convex Polygonal Mobile Robot in 2-d Polygonal Space* Proceeding of the 4th ACM Symposium on Computational Geometry, pp. 329–340, 1988

[6] K. Kant, S.W. Zucker *Toward efficient trajectory planning: The Path-Velocity Decomposition* International Journal of Robotics Research, Vol. 5, pp. 72–89, 1986

[7] T. Lozano-Perez, M. Wesley *An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles* Communications of the ACM, Vol. 22, pp. 560–570, 1979

[8] J. Pearl *Heuristics* Addison-Wesley, 1984

[9] P. Švestka, M.H. Overmars *Coordinated Motion Planning for Multiple Car-Like Robots using Probabilistic Roadmaps* IEEE Int. Conf. on Robotics and Automation, pp. 1631 – 1636, 1995

[10] J.T. Schwartz, M. Sharir *On the 'Piano Movers' Problem. II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds* Advances in Applied Mathematics, Vol. 4, pp. 298–351, 1983

[11] E. Schömer, Ch. Thiel *Efficient Collision Detection for Moving Polyhedra* Proceedings of the 11th ACM Symposium on Computational Geometry, 1995, pp. 51–60

[12] G. Wilfong *Motion Planning in the Presence of Movable Obstacles* Annals of Mathematics and Artificial Intelligence, Vol. 3, pp. 131–150, 1991

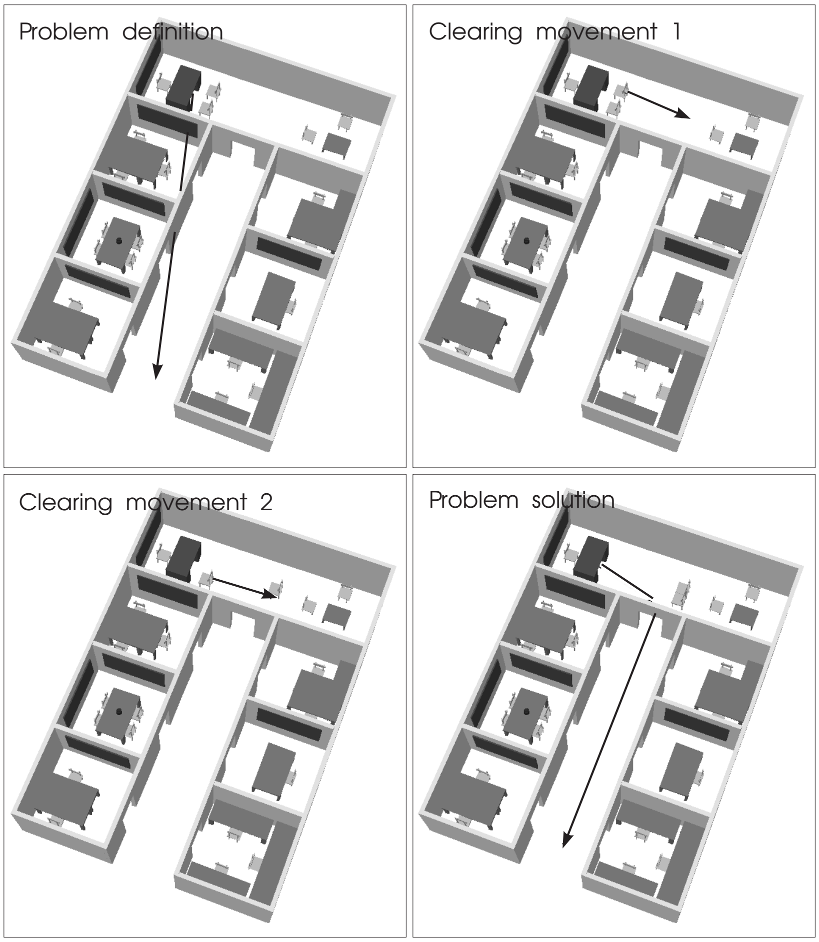

Figure 1: Solution is possible without motion coordination

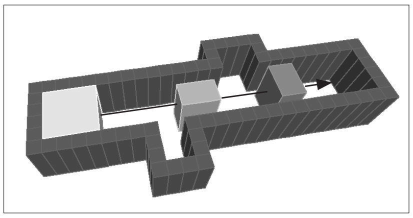

Figure 2: Solution is possible with motion ordering

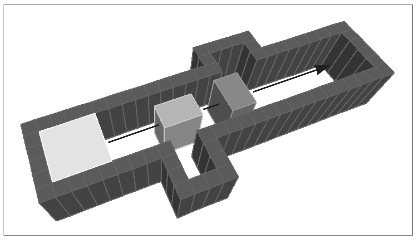

Figure 3: Solution is impossible with motion ordering (requires spatial coordination)

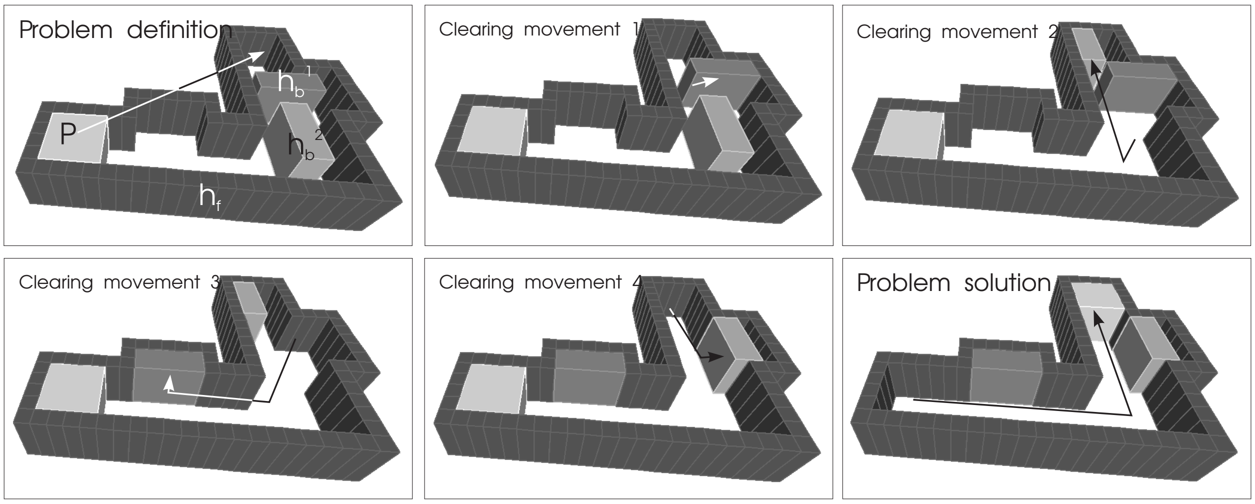

Figure 4: Solution is only possible with spatial and chronological coordination